Analyse et développement d'algorithmes parallèles pour la résolution directe de grands systèmes linéaires creux

Ibrahima GUEYE[1], Xavier JUVIGNY[1], François-Xavier ROUX[1], Frédéric FEYEL[1] & Georges CAILLETAUD[2]

[1]*ONERA*
Centre de CHÂTILLON
29, avenue de la Division Lecrec 92322 CHÂTILLON Cedex, France.
Prenom.Nom@onera.fr
[2]*ENSMP*
Centre des Matériaux P. M. FOURT, ARMINES - UMR CNRS 7633
B.P. 87 91003 EVRY Cedex, France.
Georges.Cailletaud@ensmp.fr

Résumé :

La résolution directe de grands systèmes linéaires creux est un point crucial dans de nombreuses applications industrielles et scientifiques. Dans ce papier nous proposons d'introduire une factorisation parallèle des matrices dans la résolution locale de la méthode FETI. Pour cela, nous mettons au point un solveur direct parallèle efficace basé sur la technique de dissection emboîtée des matrices. Cette technique permet de diviser la factorisation d'une matrice en autant d'étapes que de niveaux de la dissection. Ce solveur permettra d'inverser de grands systèmes linéaires et de détecter localement des singularités en cas de présence de modes rigides dans certains sous-domaines. Nous présentons quelques résultats obtenus montrant que le solveur mis au point atteint des performances respectables. Les performances de la méthode FETI utilisant dans sa résolution locale ce solveur sont aussi analysées.

Abstract :

Direct resolution of large sparse linear systems is a crucial point in many industrial and scientific applications. In this paper we propose to introduce a parallel factorization of the matrix into the local resolution of the FETI method. For that, we develop an effective parallel direct solver based on the nested dissection of the matrix. This technique makes it possible to divide the factorization of a matrix into as many stages as of levels of the dissection. This solver will allow to invert large linear systems and to detect locally singularities in the event of presence of rigid modes in some sub-domains. We present some results obtained showing that the solver developped reaches sizeable performances. The performances of the FETI method using in its local resolution this solver are also analyzed.

Mots-clefs :

dissection emboîtée ; solveur direct parallèle ; méthode FETI

1 Introduction

La résolution directe de systèmes linéaires creux constitue une base essentielle pour la simulation numérique de nombreux problèmes de calcul scientifique, notamment les problèmes liés aux calculs des structures mécaniques. En mécanique des structures, l'utilisation des codes de calcul par éléments finis conduit à des systèmes linéaires de très grandes tailles dont la résolution mène souvent à des coûts très importants en temps CPU et en espace mémoire. La parallélisation devient alors une technique incontournable, si l'on veut résoudre en implicite des problèmes de plusieurs millions d'équations avec la nécessité de prendre en compte des composants de plus en plus complexes voire des systèmes entiers. Les méthodes de décomposition de

domaine sans recouvrement sont alors un moyen naturel permettant d'arriver à réduire ces coûts de calcul. Une des méthodes les plus utilisées est la méthodes FETI[1] [2, 7]. Elle repose sur une "approche duale" en introduisant des conditions de raccord en contraintes aux interfaces entre sous-domaines. Si nous voulons conserver un nombre raisonnable de sous-domaines (quelques centaines) nous serons malgré tout confronté à des systèmes locaux de tailles importantes sur de très gros modèles (dizaines de millions d'inconnues). Pour contourner ce problème et tirer profit des nouvelles machines multi-cœurs, nous nous orientons vers un traitement multi-échelles respectant leur topologie particulière. Nous utilisons alors la méthode FETI classique et nous introduisons dans l'inversion des systèmes locaux une factorisation parallèle des matrices. Nous mettons donc au point un solveur direct parallèle efficace reposant sur la technique de dissection emboîtée des matrices [1, 3, 4]. Cette technique permet de renuméroter les matrices et de diviser la factorisation en autant d'étapes que de niveaux de la dissection.

Dans ce papier nous allons d'abord parler de la réalisation du solveur. Nous discuterons de la technique de dissection emboîtée utilisée ainsi que sa mise en œuvre. Nous présenterons ensuite quelques résultats collectés montrant que notre solveur atteint des performances comparables au solveur direct DSCPACK[2]. Les performances de la méthode FETI utilisant dans sa résolution locale le solveur DSCPACK [6] et le notre sont aussi analysées.

2 Réalisation du solveur

Il existe dans le commerce de nombreux solveurs directs parallèles, mais aucun ou du moins peu savent détecter automatiquement les singularités de la matrice décrivant le système linéaire à inverser. Ces singularités sont dûes à la présence de modes rigides dans les structures à traiter. Ici nous mettons en œuvre un solveur direct parallèle basé sur la technique de dissection emboîtée. En cas de présence de modes rigides dans les structures, nous rajoutons une étape pour le traitement spécifique des singularités.

2.1 La dissection emboîtée

Les techniques de renumérotation des inconnues des systèmes linéaires creux visent essentiellement à réduire le nombre d'opérations et à minimiser le remplissage des matrices pendant la phase de factorisation. Ce qui permet de réduire le temps nécessaire à la factorisation ainsi que la taille mémoire à utiliser. Une des approches efficaces de renumérotation est la dissection emboîtée. Pour développer le solveur nous effectuons une dissection emboîtée incompléte. L'approche que nous avons choisie consiste à renuméroter par groupe d'inconnues (sous-domaines) au lieu de faire une renumérotation inconnue par inconnue.

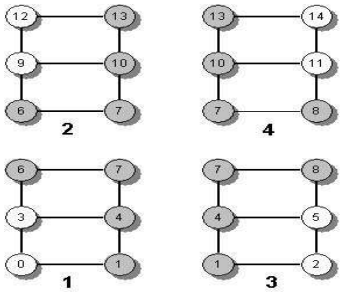

FIG. 1 – Le découpage en quatre blocs.

[1] Finite Element Tearing and Interconnecting
[2] Domain Separator Codes Package

Pour illustrer cette technique, nous considérons un système linéaire à résoudre $Mx = b$, où M est une matrice de taille 3x5 issue de la discrétisation par différences finies du Laplacien en dimension deux. Nous découpons en quatre sous-domaines le graphe associé à la matrice M (voir figure 1). La technique consiste à renuméroter sous-domaine par sous-domaine les inconnues du système linéaire. A l'aide d'un algorithme génétique nous reconstituons l'arbre d'élimination des inconnues tout en minimisant le nombre d'inconnues interconnectées aux interfaces (figure 2(a)). L'élimination des inconnues par sous-domaine permet alors de ne pas créer de nouvelles dépendances transversales (de nœuds de même niveau) sur l'arbre. En revanche, le calcul des compléments de Schur des différents sous-domaines va modifier certaines des lignes et colonnes des sous-blocs situés plus bas dans l'arbre. Pendant la factorisation, cette technique permet aussi de ne pas remplir énormément le profil de la matrice M renumérotée dont la structure par blocs est présentée sur la figure 2(b). Avec un tel arbre d'élimination la matrice factorisée aura la même structure que la matrice renumérotée.

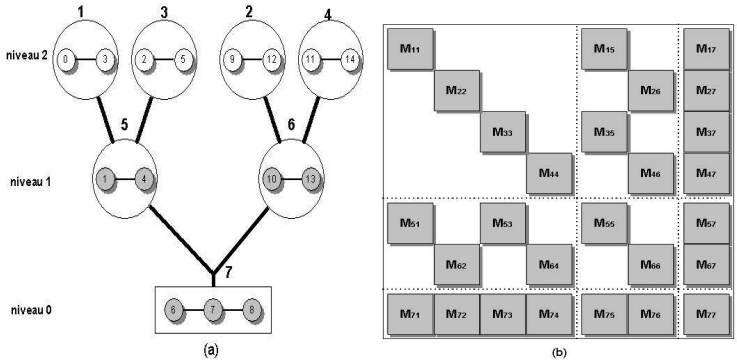

FIG. 2 – L'arbre et matrice renumérotée obtenus par dissection.

2.2 La mise en œuvre

Avec la nouvelle structure de la matrice M, pour résoudre le système linéaire nous effectuons une factorisation par niveaux en commençant par le plus haut niveau de l'arbre (le niveau 2). Cette première étape constitue la phase de factorisation locale des blocs M_{ii}, $i = 1, 2, 3$ et 4. L'élimination des inconnues associées à ces blocs apporte des contributions $M_{fi}M_{ii}^{-1}M_{ig}$ aux sous-blocs M_{fg}, pour f, g prenant les valeurs 5, 6 et 7. Par contre, aucun terme non nul ne peut apparaître au cours de la factorisation locale entre les inconnues d'un bloc M_{ii} et les inconnues d'un bloc M_{jj}, pour $i \neq j$ prenant les valeurs 1, 2, 3 et 4. Le calcul du complément de Schur sur les blocs 5 et 6 situés au niveau suivant (niveau 1) donne respectivement :

$$\begin{pmatrix} S_{55} & S_{57} \\ S_{75} & 0 \end{pmatrix} =$$

$$\begin{pmatrix} M_{55} - M_{51}M_{11}^{-1}M_{15} - M_{53}M_{33}^{-1}M_{35} & M_{57} - M_{51}M_{11}^{-1}M_{17} - M_{53}M_{33}^{-1}M_{37} \\ M_{75} - M_{71}M_{11}^{-1}M_{15} - M_{73}M_{33}^{-1}M_{35} & 0 \end{pmatrix} \quad (1)$$

et

$$\begin{pmatrix} S_{66} & S_{67} \\ S_{76} & 0 \end{pmatrix} =$$

$$\begin{pmatrix} M_{66} - M_{62}M_{22}^{-1}M_{26} - M_{64}M_{44}^{-1}M_{46} & M_{67} - M_{62}M_{22}^{-1}M_{27} - M_{64}M_{44}^{-1}M_{47} \\ M_{76} - M_{72}M_{22}^{-1}M_{26} - M_{74}M_{44}^{-1}M_{46} & 0 \end{pmatrix} \quad (2)$$

Le calcul du complément de Schur sur le bloc M_{77} situé au niveau 0 de l'arbre va nécessiter les contributions $M_{7i}M_{ii}^{-1}M_{i7}$ des blocs M_{ii}, $i = 1$, 2, 3 et 4. On assemble donc S_{77} sous la forme :

$$S_{77} = M_{77} - \sum_{i=1}^{4} M_{7i}M_{ii}^{-1}M_{i7} \tag{3}$$

Les blocs S_{ij} ont les mêmes dimensions que les blocs M_{ij}, i, j = 5, 6 et 7. Pour gagner en espace mémoire, l'idée est de stocker les coefficients de S_{ij} dans le bloc M_{ij}. Après la factorisation locale, ces blocs sont alors denses puisque le complément de Schur de chaque sous-domaine est une matrice pleine. Dans la phase de factorisation du complément de Schur, l'élimination des inconnues des interfaces 5 et 6 apporte des contributions au complément de Schur sur le bloc M_{77} (équation 4). Pour la réalisation de la factorisation de façon efficace, il est nécessaire d'utiliser des routines de calcul par blocs de type BLAS 3, sur le calcul des contributions des interfaces.

$$S_{77} = S_{77} - \sum_{i=5}^{6} S_{7i}S_{ii}^{-1}S_{i7} \tag{4}$$

Lors de la factorisation, il arrive qu'on trouve une singularité locale à un bloc matriciel diagonal. Cette singularité peut n'être qu'un effet indésirable dû au découpage ou une singularité de la matrice globale. Ainsi, lorsqu'on rencontre durant la factorisation une singularité locale, on la rajoute à une liste de nœuds singuliers. A la fin de la factorisation, on essaye d'éliminer ces nœuds en complétant globalement la factorisation de la matrice à l'aide du complément de Schur qu'on factorise par la suite. Les singularités détectées lors de la factorisation du complément de Schur donnent les singularités globales et celles qui n'étaient que locales sont éliminées.

La structure particulière de la matrice permet une résolution par blocs du système $Mx = b$. L'idée est d'écrire M sous la forme LU où L et U sont définies respectivement par les matrices triangulaires ci-dessous :

$$\begin{pmatrix} M_{11} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & M_{22} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & M_{33} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & M_{44} & 0 & 0 & 0 \\ M_{51} & 0 & M_{53} & 0 & S_{55} & 0 & 0 \\ 0 & M_{62} & 0 & M_{64} & 0 & S_{66} & 0 \\ M_{71} & M_{72} & M_{73} & M_{74} & S_{75} & S_{76} & I \end{pmatrix} et \begin{pmatrix} I & 0 & 0 & 0 & M_{11}^{-1}M_{15} & 0 & M_{11}^{-1}M_{17} \\ 0 & I & 0 & 0 & 0 & M_{22}^{-1}M_{26} & M_{22}^{-1}M_{27} \\ 0 & 0 & I & 0 & M_{33}^{-1}M_{35} & 0 & M_{33}^{-1}M_{37} \\ 0 & 0 & 0 & I & 0 & M_{44}^{-1}M_{46} & M_{44}^{-1}M_{47} \\ 0 & 0 & 0 & 0 & I & 0 & S_{55}^{-1}S_{57} \\ 0 & 0 & 0 & 0 & 0 & I & S_{66}^{-1}S_{67} \\ 0 & 0 & 0 & 0 & 0 & 0 & S_{77} \end{pmatrix}$$

La solution du système est obtenue en resolvant les systemes triangulaires $Ly = b$ et $Ux = y$. La résolution de ces systèmes triangulaires s'effectue en trois grandes phases :

– une phase de descente ; résolution des systèmes locaux $M_{ii}y_i = b_i$, $i = 1$, 2, 3 et 4 et mise à jour des termes b_i $i = 5$, 6 et 7 du second membre b :

$$\begin{pmatrix} b_5 \\ b_6 \\ b_7 \end{pmatrix} = \begin{pmatrix} b_5 \\ b_6 \\ b_7 \end{pmatrix} - \begin{pmatrix} M_{51}y_1 + M_{53}y_3 \\ M_{62}y_2 + M_{64}y_4 \\ M_{71}y_1 + M_{72}y_2 + M_{73}y_3 + M_{74}y_4 \end{pmatrix} \tag{5}$$

– une phase de résolution du problème condensé aux interfaces :

$$\begin{pmatrix} S_{55} & 0 & 0 \\ 0 & S_{66} & 0 \\ S_{75} & S_{76} & I \end{pmatrix} \begin{pmatrix} I & 0 & S_{55}^{-1}S_{57} \\ 0 & I & S_{66}^{-1}S_{67} \\ 0 & 0 & S_{77} \end{pmatrix} \begin{pmatrix} x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} b_5 \\ b_6 \\ b_7 \end{pmatrix} \tag{6}$$

– une phase de remontée ; calcul des solutions locales :

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} - \begin{pmatrix} M_{11}^{-1}M_{15}x_5 + M_{11}^{-1}M_{17}x_7 \\ M_{22}^{-1}M_{26}x_6 + M_{22}^{-1}M_{27}x_7 \\ M_{33}^{-1}M_{35}x_5 + M_{33}^{-1}M_{37}x_7 \\ M_{44}^{-1}M_{46}x_6 + M_{44}^{-1}M_{47}x_7 \end{pmatrix} \tag{7}$$

Avec la dissection emboîtée la parallélisation du solveur se fait de façon naturelle. Etant donné qu'il n'existe pas de dépendances sur les inconnues des sous-domaines situés à un même niveau sur l'arbre, on traite ces sous-domaines en même temps sur des ensembles séparés de processeurs. La parallélisation se fait sous l'environnement MPI.

3 Résultats et analyse

Nous présentons ici quelques résultats obtenus avec le solveur mis en place (DISSECTION). On fait une comparaison de ses résultats avec ceux des solveurs Sparse Direct (développé au CdM[3]) et DSCPACK dont la mise en œuvre est basée sur l'idée de diviser la matrice creuse en domaines et en séparateurs. Les solveurs Sparse Direct et DSCPACK inversent des systèmes linéaires symétriques. Les performances de la méthode FETI utilisant dans sa résolution locale ces deux solveurs sont aussi analysées.

3.1 Analyse des résultats du solveur

Actuellement, la structure du complément de Schur est considérée comme étant non symétrique. Nous avons donc fait une factorisation et une résolution non symétriques du problème condensé aux interfaces. Les jeux de tests utilisés ici portent sur des problèmes d'élasticité linéaire quadratique en 3D. Ces tests ont été effectués sur une machine bi-processeurs AMD Opteron 64-bit qui posséde 4 Go de mémoire vive. Les résultats sont présentés sur le tableau 1.

Taille du problème / **Solveur**	14739 ddls	56355 ddls	110221 ddls
DSCPACK	2.98 s	52 s	169 s
DISSECTION	5.56 s	122 s	485 s
Sparse Direct	20.26 s	427 s	1518 s

TAB. 1 – Temps d'exécution en secondes pour chaque type de solveur

Ce tableau montre que les résultats obtenus avec le solveur mis en œuvre dans sa version non symétrique sont encourageants. Une fois la version symétrique mise en place nous pouvons amélioré ces résultats en divisant les temps de calcul par deux. Ce qui nous permettra d'égaler les temps d'exécution du solveur DSCPACK.

3.2 Analyse des performances de la méthode FETI

L'implantation de la méthode FETI dans le code de calcul par éléments finis ZeBuLon (développé conjointement par l'ONERA, le centre des Matériaux et Northwest Numerics (Seatle)) permet la résolution en parallèle de problèmes à très grands nombres de degrés de liberté. La

[3]Centre des Matériaux

méthode telle qu'elle est mise en œuvre aujourd'hui dans le code peut utiliser le solveur DSCPACK pour inverser les systèmes locaux. On veut utiliser aussi la version symétrique du solveur DISSECTION dans la méthode FETI pour comparer les deux résultats.

Nous avons réalisé des expérimentations parallèles en utilisant le solveur DSCPACK dans la résolution locale de FETI. Les calculs sont faits sur un cluster Linux doté de 78 nœuds biprocesseurs AMD Opteron disposant chacun de 4, 8 ou 16 Go de mémoire vive. Les nœuds communiquent des données au travers d'un réseaux Gigabit Ethernet. Ils nous reste à faire de même avec le solveur mis en place pour pouvoir faire la comparaison. Les résultats de comparaison seront présentés prochainement.

4 Conclusions

Les résultats obtenus avec le solveur mis en œuvre dans sa version non symétrique sont encourageants. Des travaux sur la version symétrique du solveur sont en cours de réalisation. Ces résultats peuvent être améliorés une fois la version symétrique prête. Nous nous attendons donc à diviser les temps de calcul par deux ce qui nous permettra d'atteindre des performances respectables. La parallélisation du solveur va bientôt être commencée. Dès la fin de la parallélisation, nous pourrons faire du multi-domaines dans la résolution locale de la méthode FETI (factorisation parallèle des systèmes locaux). L'utilisation de la version parallèle du solveur dans la méthode FETI nous permettra de tester la détection automatique de singularités en cas de présence de modes rigides dans certains sous-domaines.

Références

[1] P. Charrier and J. Roman. Algorithmique et calculs de complexité pour un solveur de type dissections emboîtées. *Numerische Mathematik*, 55 :463–476, 1989.

[2] C. Farhat and F.-X. Roux. Implicit parallel processing in structural mechanics. *Computational Mechanics Advances*, 2(1) :1–124, 1994.

[3] A. George. Nested dissection of a regular finite element mesh. *SIAM J. on Numerical Analysis*, 10(2) :345–363, 1973.

[4] A. George and J. W.-H. Liu. *Computer solution of large sparse positive definite systems*. Prentice Hall, 1981.

[5] P. Lascaux and R. Théodor. *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, volume 1. Masson, 1986.

[6] P. Raghavan. Dscpack home page. *http ://www.cse.psu.edu/ raghavan/Dscpack*, 2001.

[7] D. J. Rixen. Substructuring and dual methods in structural analysis. *Thèse de doctorat, Université de Liége*, 1997.